1 Introduction

Embedded systems advance to multi-core
• still limited resources (power, memory, clock-rate)
• shrinking feature size and increasing error rate

Dependable embedded systems
• safety or mission critical
• hard real-time and dependability requirements

Transactional memory (TM) provides best
effort optimistic concurrency control
• ease programming with high scalability potential
• execute in isolation & either commit atomically or
roll back all changes to prior consistent state

Failure control for improved dependability
• use TM for failure isolation
• failure atomicity: failed operations leave target
component in state prior to the invocation

2 Previous Work

VELOX Integrated TM Stack (EU FP7)
Dresden TM Compiler [1]
• C/C++ language integration of atomic blocks
• instrumentation of atomic blocks using LLVM pass

TINYSTM [2,3,4]
• word-based TM with lazy snapshot algorithm

ROBUSSTM [5]
• word-based TM with practically wait-free liveness
• tolerates crashed and non-terminating threads

[1] Riegel et. al: Evaluation of AMD's Advanced Synchronization Facility
Within a Complete Transactional Memory Stack; EUROSYS 2010
Software Transactional Memory; PPoPP 2008
[3] Riegel, Fetzer, Felber: Time-based Transactional Memory with Scalable
Time Bases; SPAA 2007
DISC 2006
Transactional Memory; SSS 2010

3 CHALLENGE I: Failure Isolation of Tasks

Dependability requires strict spatial,
temporal, and failure isolation
• tasks communicate via shared memory that is
protected by TM
• faulty or less critical tasks must not negatively
affect other tasks
• TM enforces
• consistency: operations execute isolated &
updates in shared region become visible
atomically (commit) or all changes are rolled
back (abort)
• liveness: fairness among tasks & timely
resource usage

4 CHALLENGE II: Real-Time Constraints

Extend TM language primitives and runtime
systems to satisfy real-time constraints
• TM does not suffer from priority inversion
compared to lock-based synchronization
• TM must provide real-time and access guarantees
• memory access only permitted in predetermined
time slots & roll back unfinished transactions

5 CHALLENGE III: SW and HW Faults

Ensure consistency in the face of software
and hardware faults
• check executable constraints (invariants) before
committing updates: change transaction
demarcation
• begin transaction at program start, after commit
or return from system call (external action)
• commit transaction when control flow reaches
executable assertion, system call or termination

6 CHALLENGE IV: Replication Mechanisms

Integration with replication mechanisms
• active replication executes each operation on a
set of processes for instant fail-over
• scheduling of threads introduces non-
determinism that causes replicas to diverge
• TM must integrate a form of deterministic
execution scheduling or roll back after
divergence was detected
• time-triggered TM ensures determinism without
additional overhead